5: CONTENTS OF DISK

A disk contains the following 3 directories:

1: \USER
This directory contains the files that may be used in the user program.

I2C_DR.LIB	Library with I²C routines.
I2C.H	Header file for C applications.
I2C.MAC	Macro's for the I²C routine calls in assembly programs.
VAR_DEF.ASM	Include file with variable definitions for assembly programs.
EXT_VAR.ASM	Include file with external definitions for assembly programs.
LIB.BAT	Example batch file to create I2C_DR.LIB.
ASM.BAT	Examplebatch file to assemble source modules for library.

2: \EXAMPLE
This directory contains the source files of the examples described in chapter 4.

DEMO_ASM.*	Assembly example.
DEMO_PLM.*	PL/M example.
HEAD_51.SRC	Example of environment file for PL/M example.
DEMO_C.*	C example.
CSTART.ASM	Example of environment file for C example.

3: \SOURCE
This directory contains the source files of the modules in the library.

Using the P82B715 I²C extender on long cables AN444

Author: Don Sherman, Sunnyvale

The P82B715 I²C Buffer was designed to extend the range of the local I²C bus out to 50 Meters. This application note describes the results of testing the buffer on several different types of cables to determine the maximum operating distances possible. The results are summarized in a table for easy reference.

The I²C bus was originally conceived as a convenient 2 wire communication method between Integrated Circuits located within a common chassis, such as inside a TV set or inside a VCR. The serial protocol contains an address, or identifying code, for each type of device and additional internal addresses, if needed within the addressed device. Each device has its own decoding circuitry to allow it to recognize its own unique address or identifying code. To communicate, a device watches the bus activity and jumps in when it sees a stop. Once a Master gets control of the bus, it sends the address of the particular device with which it wants to communicate. Communication will then transpire between the Master and the Slave device. The existence of many types of ICs which have built-in I²C interface capabilities makes system design almost as easy as drawing a block diagram. Real-time clocks, RAM, A/D converters, EEPROMs, Microcontrollers, Keyboard encoders, LCD display drivers, and many other I²C supported chips all communicate over two wires rather than needing 16 Address lines, 8 data lines and Address decoders along with handshake signals, which more conventional designs would require to be routed all over the Printed Circuit board.

Now, with the introduction of the I²C buffer chip, it is easy to branch out beyond the single chassis mode and use this convenient local area network to tie together whole systems without the need to convert from the "internal" I²C protocol to an external communication medium such as RS-232 and then RS-485. By using the new Philips I²C buffer, the external systems' components can be accessed as easily as the internal I²C connected components.

The P82B715 is an 8 pin IC which contains 2 identical amplifier sections to allow for the current amplification and buffering of both the SDA and the SCL signals on the I²C bus. Each section in the P82B715 contains a bipolar times 10 current amplifier which senses the direction of current flow through an internal 30 ohm series resistor in the I²C line. The P82B715 then boosts the current, while keeping the voltage gain at unity, and continues to maintain the voltage drop direction across the resistor. This configuration results in different waveforms as the P82B715 starts to do its job. If the driving source has a strong current sink capability, then it will start to drive the buffered I²C line immediately through the 30 ohm resistor. A microsecond later the P82B715's amplified pull down current kicks in and pulls the line down even harder. If the driving IC is only capable of the I²C specified 3 milliamp pull down current, the buffered bus will fall a little and then just wait at that voltage level for the propagation delay of the amplifier to finally turn on and bring the buffered bus down to a logic low. Thus, there will always be some form of a step in the falling edge of the buffered output waveform, see Figure 1. A weak source will have a step (plateau) up near 4 volts and a strong source, such as the Philips Semiconductors 87C751 microcontroller, will have the step occur below 2 volts. The position of the step will be determined by the current sink capability of the I²C bus driver versus the value of the pull-up resistor which is used on the buffered I²C bus, Vstep = 5V – (Isink × Rbuf). For example: Vstep = 5V – (3mA × .165 k ohms) = 5 – .495 = 4.5Volts; another example: Vstep = 5V – (20mA × .165 k ohms) = 5 – 3.3 = 1.7Volts.

Running the I²C signals over long distances poses several problems. The I²C SDA and SCL lines are monitored by all of the ICs connected on the I²C bus. These ICs each have their own circuitry to decipher the information on the bus. In normal operation, a Start occurs when there is a high to low transition on the SDA line while SCL is high. Obviously, if any external noise is coupled into the SDA line, it could be mistakenly perceived as a Start. Because of this, some form of shielding will be preferred to protect the two I²C signals from external noise sources. During the transmission of data there are signals which are active on both SDA and SCL. If these normal signals are cross-coupled, then data can be corrupted. Thus, although the standard telephone twisted pair cable is the most commonly available built in cable, it is not recommended for long I²C runs. This cable maximizes crosstalk, due to the twisted pair configuration and, since there is no shielding, is very vulnerable to adjacent wire telephone signal coupling and to any stray external electromagnetic interference. This effect can be somewhat reduced by running a signal wire and a grounded wire as adjacent pairs.

Long distance cables present capacitive loading which must be overcome with the driver chips. The limiting factor is the amount of pull-up current which is available to charge the line capacitance. With the simple resistor pull-up recommended by I²C standards, three milliamps is available for charging this line capacitance. The rise time of the signal will increase linearly with the increase in capacitive loading and the specified maximum capacitive loading is only 400 Pico Farads for guaranteed 100kHz communication rates. The P82B715 current buffer allows for 30 milliamps of pull-up current, with a resulting maximum capacitive loading of 4,000 Pico Farads (4 Nano Farads).

The I²C hardware inputs look at the I²C signals and act when those signals pass through the active linear region at about 1.2 to 1.4 volts, and are detected as digital levels. Thus, there is a delay between when an output transistor turns off and when the rising signal is detected as a logic one at the receiver. This time depends on the value of the pull-up resistor, the perceived capacitance at the transmitting end, the delay through the cable, and finally the delay through the receiver's amplifier to its output stage. The maximum allowable time is limited by the characteristic that the I²C master provides the clock signal which must travel down the cable and be received by the slave. This slave must act on the clock signal and produce data information which is sent back to the master with an additional set of delays. Upon reception the data must be put in its proper place before the master starts its next clock signal, or an error will occur.

Different types of cable were tested and the results are shown in Table 1. Keep in mind that the results are based on cable runs in a low electrical noise environment. If reliable operation is desired in a high electrical noise environment, shielded cable must be used. For "short" runs, flat cable with every other conductor grounded, seems to provide a good, low capacitance medium for I²C transmission, otherwise, the shielded audio cable seemed to provide the best price/performance. Note that for long runs, it is desirable to have a separate power supply at each end of the cable, and the shield or ground wire will provide a common reference between the two supplies. The voltage drop due to the resistance of the wire usually is the limiting factor for very long runs of cable where the power to the remote system must also come through the cable. Table 1 shows the results of testing with longer and longer cable lengths until failures were detected. The values in the table represent the maximum cable lengths which still provided error free code from a modified version of the Ping-pong program which is listed in Application Note AN430.

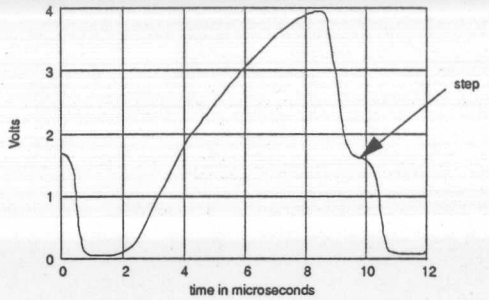


Figure 1. P82B715 Output Waveform on Long Cable

Table 1. Test Results with P82B715 Over Long Cables

CABLE TYPE	Ohms/m	pF/m	Total Length	Total Ohms	Total Cap.
Belden 8723 45 Ohm Audio 2 each 2—24AWG wire stranded Beldfoil Aluminum-polyester shielded with common drain wire SDA & ground on one pair; SCL & ground on other pair	.049	115	305M (1000')	11.5	48.2nF
Belden 8723 45 Ohm Audio using 1 shielded pair, SDA on Red, SCL on Black	.049	115	330M (1100')	12.7	53nF
RG-174/U 50 Ohm Video Cable SDA and grounded shield in one cable SCL and grounded shield in one cable	.318	101	150M (500')	47.7	15.2nF
"Telephone Cable" 22&24 AWG Solid Copper Twisted Pair, Level 3 LAN & Medium Speed Data SDA and ground in one twisted pair SCL and ground in one twisted pair	.0286	66	95M (310')	2.7	6.4nF
Flat "Ribbon" Cable, every other conductor grounded	.20	52	400M (1320')	80.5	21nF

In all of the tests, the power supply voltage was 4.5 volts. The ground for the remote test fixture was through the long cable. Since 4.5 volts is the recommended minimum voltage for both the 87C751 and the P82B715, it was not possible to operate the remote unit on power supplied through the long cable, since any ohmic drop would place the ICs out of their specified range. However, it is necessary to connect the grounds between the two units for the best noise immunity.

The P82B715 is designed to drive a 4 nF capacitive load at 100kHz. However, the actual total capacitances of the long cables which worked were substantially greater than this. The loading did effect the software driven hardware part of the 87C751. To achieve a true 100kHz data rate, it was necessary to shorten the '751 Timer values for the I²C drivers. This resulted in an asymmetrical waveform, but did achieve a 10 microsecond period (100kHz). This asymmetry in duty cycle can be easily seen in the Figure 1 waveform.

The test with the Belden 8723 Audio Cable worked if one of the shielded pair was connected to a signal and the other was connected to ground or +5volts. When both wires were connected in parallel as signal wires, the capacitance to ground doubled and the test failed. Also note that the adjacent wire mutual inductive coupling of the SDA and SCL signals did not seem to cause any problems even out to 1000 feet. This indicated that possibly the Belden 9452 45 ohm beldfoil shielded audio cable with a single set of twisted pair wires would be a good candidate to also try.

Flat ribbon cable provided a good compromise between shielding and reasonable capacitance. It is possible to increase the shielding effect by using flat cable with an etched copper foil layer on the back side of the cable. Noise can be induced into the cable by folding it back over itself for mutual induction effects, and also by operating a noise source close to the cable. A transformer type of soldering iron and florescent light transformers seemed to be good noise sources.

The P82B715 can drive multiple P82B715 remote units. The line should have some form of pull-up resistor at each driver. If only two drivers are used, as shown in Figure 2, the load should be split between the two drivers. For example, if the pull-up current is to be 30 milliamps and the voltage is 5 volts, the pull-up resistance should be: 5V/.030 amps = 165 ohms. This should be implemented by placing a 330 ohm resistor at each end of the cable so that the parallel resistance is 165 ohms and each end of the line is terminated. Remembering that the current gain can be as low as 8 and that most runs will not be to the maximum possible distance, lower values of pull-up current can be used with the appropriate modifications to the above equations.

For larger fan-out with fixed locations, the load resistance should also be evenly divided so that the parallel combination of all of the pull-up resistors will provide the desired D.C. pull-up current.

If some of the remote units will be pluggable, it will be necessary to divide the pull-up load to accommodate all of the possible combinations of possible fanout. Figure 3 shows an example of driving up to 30 remote, pluggable peripherals. On the 3 milliamp side of the P82B715 a complete I²C system may exist. In Figure 3, a local I²C network cluster could be joined to other local network clusters through the P82B715 buffered bus so that hundreds of I²C devices could potentially be interconnected.

The ease of connecting I²C clusters into a complete LAN opens the door for many new uses of components which have an I²C bus connection. Now an electronic instrument can have access to remote keyboards and remote sensors by using the I²C bus. The instrument's output can easily be shown on multiple remote displays all connected with the I²C bus. Multiple instruments can also pass data back and forth over the I²C bus. Thus, we see that the I²C bus can become an effective and inexpensive Local Area Network by using the P82B715 I²C bus extender.

THE TEST SETUP

These tests were run on two identical test boards which each use a Philips Semiconductors 87C751 microcontroller that drives the I²C buffer which has a 330 ohm pull-up resistor. The schematic is shown in Figure 4. The software is a modified version of the "Ping-Pong" program which is described in the Philips Semiconductors Application Note, AN430, "Using the 8XC751/752 in Multimaster applications". This program sends a number down the I²C line and, when received, the receiving unit becomes a master and increments the number and sends it back to the first unit where it is checked and then the process repeats itself. The software has extensive error detection capability and monitors for corruption of data, false starts, over run of data, stuck lines and about anything else which might indicate a problem. If any errors did occur, a software counter was incremented. In this setup, the counter was stopped at Hex 07F to prevent wrap around and the contents of the counter are displayed on a bank of 8 LEDs. The MSB of the counter register was used as an indicator that the unit was working. The MSB LED flashes at about a 1 Hz rate when the unit is operating normally. When a cable length was reached which was too long, the MSB LED would stop flashing and the counter would rapidly fill up and stop with all 7 LEDs on (LED on indicates a logic "1" in this application).

THE TEST HARDWARE

A general purpose test rig was designed so that future needs of a general I²C platform could also be met. All of the port pins on the '751 were used. The inputs to the system were a toggle switch with a pull-up resistor connected to P0.2 (because this pin is Open Drain) and an octal DIP switch connected to port 1 (the internal pull ups of the port were used, so no external pull-up resistors were needed). The output is displayed through an octal buffer connected to port 3. A logical "1" on the pin will light up the LED. The I²C signals, SDA and SCL, are connected to the I²C buffer chip and the outputs of the buffer are pulled up by 330 ohm resistors. The parallel combination of the buffered transmitting end pull-up and the receiving end pull-up resistors is 330/2 ohms, which results in a pull-up load current of 30 milliamps. This current from the two pull-up resistors must be sunk by the single driving transistor of the acting sender. The effective loading seen by the '751 is the I²C buffer's load divided by 10. Thus, the '751's I²C outputs will sink 3 milliamps when driving the I²C buffer which is sinking 30 milliamps on the buffered bus.

The software monitor routine allows the user to monitor any internal '751 RAM location and display the contents on the LEDs. The monitor routine also allows the user to modify the contents of any RAM location including SFR space. The Ping-Pong program needed the first 8 locations in RAM, so the stack pointer for this application was changed from the default location of 07H to location 09H. This starts the stack at 0AH.

To read the contents of RAM, set the DIP switches to the desired RAM address. The toggle switch is set to a "1". Pressing the Reset switch causes the microprocessor to reset and then enter the monitor program where the program then waits until the toggle switch is changed. Upon closing the toggle switch (a "1" to "0" transition) the program loads the DIP switch selection into R0 of bank 1 (RAM location 08H). The program then loads the contents of the RAM location pointed to by R0 (bank 1) and copies it into port 3, where it is displayed on the 8 LEDs. Thus, the Address is seen by looking at the DIP switches and the contents pointed to are displayed on the LEDs. Note that this indirect Address latch location (R0,bank 1) would have been the normal beginning of the stack, had it not been changed.

The contents of an internal RAM location can also be modified with this program. First, set the DIP switches to the desired Address and set the toggle switch to "0". Reset the processor and then set the toggle switch to "1". This transfers the address to R0 (bank 1). Next, load the desired new data, which is to be stored in RAM, into the DIP switches, and then set the toggle switch to "0". At this time the LEDs will now show the Address of RAM and the DIP switches show what was written into the selected RAM location. To verify that the data was actually written into the RAM, follow the read RAM sequence.

Although this may seem to be a bit cumbersome, it is a workable way to see what is happening inside of the '751. Remember that it is necessary to re-enter the monitor program, or at least to duplicate the read RAM of R0 (bank 1) and output to port 3, to see the latest version of the contents of the RAM location. Since this experiment only looked at the contents of one RAM location, the above method was easy to use and the display always showed the current status of the desired RAM location because it is updated often by the software.

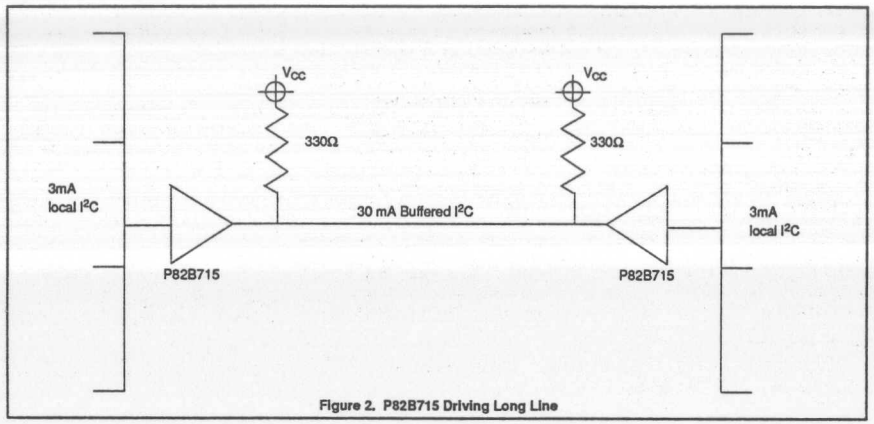
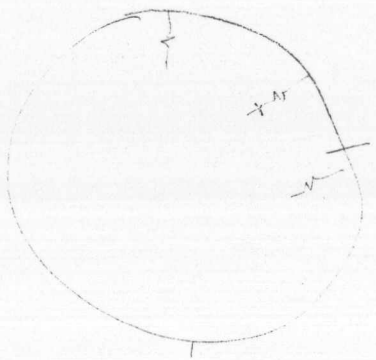


Figure 2. P82B715 Driving Long Line



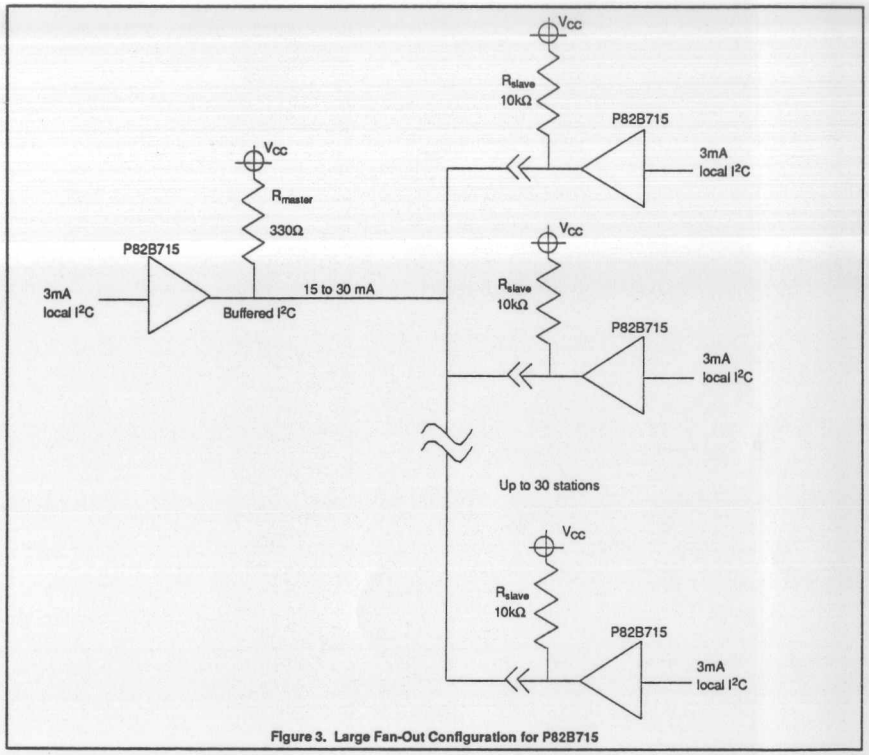


Figure 3. Large Fan-Out Configuration for P82B715

Note that V_{CC} is 5 volts for these values of load resistors. If a different voltage is desired, the calculations are as follows:

$R_{master} = \frac{V_{CC}}{15mA}$ example: $R_{master} = \frac{5V}{15mA} = 0.33k = 330\Omega$

The pluggable units would be calculated as follows:

Parallel combination of $R_{slave} = R_{master}$

$R_{slave} = R_{master} \times$ Fan out

example: $R_{slave} = 330\Omega \times 30 = 9900\Omega = 10k$

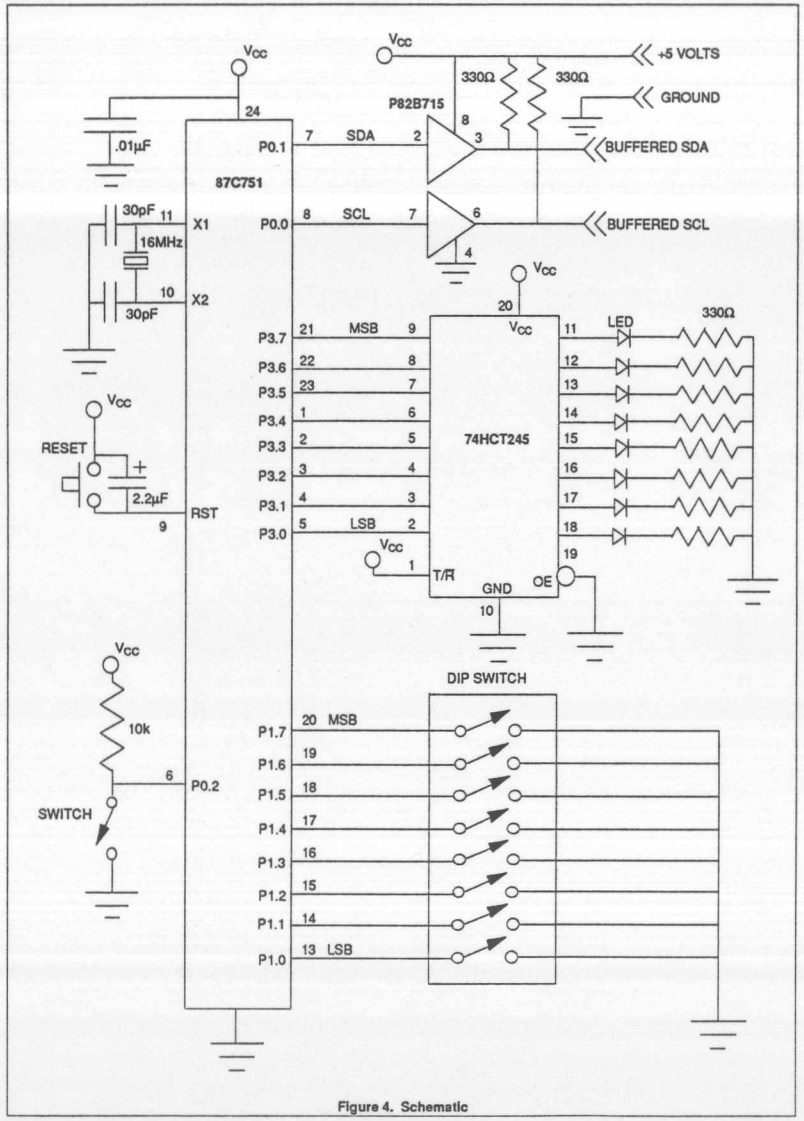


Figure 4. Schematic


```
;
;*************************************************************************
;         Multimaster Code for 83C751/83C752
;         4/14/1992    MODIFIED BY DON SHERMAN 5-21-92
;         ;; is used to show where original code was modified
;*************************************************************************
; This code was written to accompany an application note. The I2C routines
; are intended to be demonstrative and transportable into different
; application scenarios, and were NOT optimized for speed and/or memory
; utilization.
;
; Yoram Arbel

$TITLE(83C751 Multi Master I2C Routines)
$DATE(4/14/1992)
$MOD751    ;;NEED TO USE $MOD752 FOR 752 EMULATOR
;;EI2        EQU     ES          NEED ENABLE FOR EMULATOR
$DEBUG

;*************************************************************************
;       8XC751 MULTIMASTER I2C COMMUNICATIONS ROUTINES
;       Symbols and RAM definitions
;*************************************************************************

; Symbols (masks) for I2CFG bits.

BTIR        EQU     10h                 ; TIRUN bit.
BMRQ        EQU     40h                 ; MASTRQ bit.


; Symbols (masks) for I2CON bits.

BCXA        EQU     80h                 ; CXA bit.
BIDLE       EQU     40h                 ; IDLE bit.
BCDR        EQU     20h                 ; CDR bit.
BCARL       EQU     10h                 ; CARL bit.
BCSTR       EQU     08h                 ; CSTR bit.
BCSTP       EQU     04h                 ; CSTP bit.
BXSTR       EQU     02h                 ; XSTR bit.
BXSTP       EQU     01h                 ; XSTP bit.


; Note:
;
; Specific bits of the I2CON register are set by writing into this register a
; combination of the masks defined above using the MOV command.
; The SETB command should not be used with I2CON, as it is implemented by
; reading the contents of the register, setting the appropriate bit and
; writing  it back into the register.  As the functionality of the Read and
; Write  portions of the I2CON register is different, using SETB may cause
; unwanted results.


; Message transaction status indications in MSGSTAT:


SGO         EQU     10h         ; Started Slave message processing.
SRCVD       EQU     11h         ; as a slave, received a new message
SRLNG       EQU     12h         ; received as slave a message which is too
                                ; long for the buffer
STXED       EQU     13h         ; as slave, completed message transmission.
```